

## Framework PHP - AMEPE

Desenvolvido por Diogo Bezerra | [d.bezerra@yahoo.com.br](mailto:d.bezerra@yahoo.com.br)

## Sumário

1. Premissas.....	3
2. Diretórios e arquivos.....	3
2.1. Diretórios e arquivos do Sistema .....	3
2.2. Diretórios e arquivos de programação livre .....	7
3. Classes MVC .....	8
3.1. Classes Model (Modelo).....	8
3.2. Classes Views (Visão).....	8
3.3. Classes Control (Controle) .....	9
4. Seqüência de Funcionamento - index.php e .htaccess .....	9
5. Exemplos .....	10
5.1. VsExemploHome.class.php .....	10
5.2. ExemploCtrlHome.class.php .....	12
5.3. ExemploModelo.class.php.....	13
5.4. ReposExemploModelo.class.php.....	14
6. Plugins.....	14
...	14

## 1. Premissas

- ✓ PHP 5.3 ou superior (não testado com php 7).
- ✓ Conhecimentos intermediário/avançado em programação orientada a objetos.
- ✓ Conhecimento básico em arquitetura MVC.











## 2. Diretórios e arquivos


### 2.1. Diretórios e arquivos do Sistema


São diretórios utilizados pelo framework. Não devem ser apagados ou renomeados.


 **framework:** possui outros diretórios de sistema.


 **config:** possui arquivos de configuração.


-  **setBD.php:** Define as configurações de conexão com o banco.
-  **setCript.php:** Define strings para funções de criptografia.
-  **setCss.php:** Define os scripts CSS que serão utilizados.
-  **setGlobalVars:** Define um array de variáveis para configurações globais. Utilizado por Global.class.php.
-  **setManutencao:** Define se o site está ou não em manutenção. Chama a view indicada em \$VIEWMANUTENCAO caso positivo.
-  **setMeta:** Define as meta tags do site.
-  **setPlugins:** define as classes de plugins que serão utilizadas pelo site. Os plugins são classes customizadas para executar alguma função e estão localizados em [framework/plugins].
-  **setRoutes:** Define as rotas (url) para chamada de funções em classes de controle. [Ver o arquivo para mais informações].
-  **setScripts:** Define os scripts js que serão carregados para uso em todo o site. [Ver o arquivo para mais informações]
-  **setTitulo:** Define o título da aba do navegador.


 **plugins:** Armazena classes customizadas para executar alguma função.  
[Ver sessão de plugins].


 **lib:** Armazena os arquivos das classes de estrutura do framework.


 **dhtml:** Armazena todas as classes responsáveis pela geração de HTML programaticamente (no código php).


 **DTag.class.php:** É a principal classe de criação das tags HTML. É uma superclasse ou seja todas as outras classes de geração de HTML herdam de DTag.class. [Ver Exemplos].


 **html:** Classes antigas de geração de html (Deprecated).


 **sql:** Armazena as classes responsáveis pela criação e execução de queries SQL para acesso ao banco.


 **ExecuteSql.class.php:** Responsável por traduzir as queries montadas pelas demais classes de SQL e executá-las. Também executa queries passadas como string. Utiliza a classe Conexao.class.php (global/app.modelo/bd). [Ver exemplos].


 **\_autoLoadClass.php:** Arquivo responsável por incluir automaticamente as classes que são instanciadas. Obs: Somente as classes de sistema (lib/\*, AppModelo, AppView e AppCtrl) são explicitamente definidas no array \$dir. Todas as outras classes criadas em qualquer diretório, são automaticamente definidas em Global.php.class->buscaClasses().


 **ctrl.php:** Esse arquivo é chamado via Ajax para executar funções de classes de controle. As funções que chamam esse arquivo estão em global/app.controle/js/Global.js e são descritas mais adiante.

 **view.php:** Esse arquivo é chamado via Ajax para instanciar classes de visão (que iniciam com Vs). Por exemplo: mostrar uma view em uma div quando clica em um botão. A função que chama esse arquivo está implementada em global/app.controle/js/Global.js e é descrita mais adiante.


 **global:** Armazena os diretórios de modelo, visão e controle globais (MVC).

 **app.controle:** Armazena os arquivos de controle.


 **app.controle/js:** Diretório que armazena os arquivos javascript utilizados globalmente.


 **Global.js:** Possui funções responsáveis pela chamada de controles e views e envio de formulários. São funções essenciais para o funcionamento do sistema. Utilizam Ajax para carregar a resposta em um container especificado (target) (com exceção de submitFormToLink). Seguem as descrições principais.


- **callCtrlPost:** Direciona para ctrl.php instanciando uma classe de controle informada e chamando uma função desse controle passando parâmetros via post. Pode ser chamada no php com a função `AppCtrl::callCtrlPost` [Ver exemplos].
- **callCtrlPostConcat:** Possui a mesma função de `callCtrlPost`, porém concatena o resultado com o conteúdo já existente do container no qual será exibido o resultado. Pode ser chamada no php com a função `AppCtrl::callCtrlPostConcat` [Ver exemplos].
- **submitForm:** Direciona para ctrl.php instanciando uma classe de controle informada e chamando uma função desse controle passando como parâmetros os campos de um form. Pode ser chamada no php com a função `AppCtrl::submitForm` [Ver exemplos].
- **formUpload:** Possui a mesma função de `submitForm`, porém também passa arquivos para upload (`<input type="file">`). Pode ser chamada no php com a função `AppCtrl::formUpload` [Ver exemplos].
- **submitFormToLink:** Submete um formulário para um link especificado. Pode ser chamada no php com a função `AppCtrl::submitFormToLink` [Ver exemplos].
- **showVs:** Direciona para view.php instanciando uma classe de visão informada. Pode enviar parâmetros para a classe se necessário. Pode ser chamada no php com a função `AppCtrl::showVs`. [Ver exemplos].

 **onLoad.js:** Esse arquivo possui funções responsáveis pelo carregamento dos scripts definidos em `framework/config/setScripts.php`. É escrito na tag head no `index.php` através da função `loadhead` da classe `framework/lib/dhtml/DHead.class.php`.


Os demais arquivos são scripts js de funções de livre programação e podem ser incluídas em `framework/config/setScripts.php`.


 **app.controle/AppCtrl.class.php:** `AppCtrl` é uma superclasse herdada por todas as classes de controle (`Ctrl[nome]`). Ela possui as funções que montam a chamada das funções de `Global.js` para que possam ser chamadas mais facilmente através de código PHP. [Ver exemplos].


 **app.controle/CtrlFooter.class.php e CtrlHeader.class.php:** As classes de visão (`Vs[nome]`) podem ou não possuir controles. Essas classes são referentes às classes de visão `VsFooter.class.php` e `VsHeader.class.php` respectivamente, descritas mais adiante.


 **app.controle/Global.class.php:** Essa classe é instanciada no `index` e é imprescindível para o funcionamento do sistema. É responsável pela inclusão de outros arquivos essenciais como `_autoLoadClass.php`, `Conexao.class.php` e `simple_html_dom.php` e fornece a todo o sistema as variáveis globais definidas em `framework/config/setGlobalVars.php` além de informar à função `autoLoadClass` a localização das demais classes (`buscaClasses()`).


Outras funções de uso global menos importantes podem ser incluídas livremente.


 **app.modelo:** Esse diretório armazena todas as classes de modelo do sistema e seus repositórios. [Ver Classes MVC]. É onde está também o arquivo de conexão.

 **app.modelo/bd:** Armazena o arquivo de conexão `Conexao.php.class`. Essa classe é responsável pela seleção do banco e pela abertura e fechamento de conexões utilizando PDO.


 **app.model/repositorios:** Toda classe de modelo possui obrigatoriamente seu repositório correspondente. Essas classes de repositório são armazenadas nesse diretório e são responsáveis pela criação e execução das queries. Todas as classes herdam da superclasse Repos.class.php [Ver exemplos].

 **app.modelo/AppModelo.class.php:** Superclasse herdada por todas as classes modelo. Identifica, define e acessa a classe de repositório e a tabela do BD correspondentes e possui as funções get e set dos objetos.

 **app.visual:** Armazena as classes globais de visão (Vs[nome]) e arquivos globais relativos à visão do sistema. Cada classe de visão deve ter obrigatoriamente um arquivo html correspondente e pode ou não possuir um arquivo css e um arquivo de script js. [Ver Classes MVC].

 **app.visual/AppView.class.php:** AppView.class.php é uma super classe herdada por todas as classes de visão. É responsável por definir os arquivos correspondentes html, css, js, header e footer utilizados pela classe. Possui funções de obtenção e manipulação de tags do arquivo html correspondente utilizando simple\_dom\_html (modificado) além de outras. [Ver Classes MVC].

 **app.visual/Vs404.class.php:** Classe view para erros 404.

 **VsFooter.class.php e VsHeader.class.php:** Classes de visão onde podem ser implementados o jeader e o footer global do site. Podem ou não serem utilizadas por qualquer view. [Ver Classes MVC].

## 2.2. Diretórios e arquivos de programação livre

Qualquer diretório pode ser criado para armazenar as classes que serão programadas para montar o site. O framework se encarrega de identificar as classes em qualquer desses diretórios. Obs: Deve-se ter cuidado para não criar classes com nomes iguais.

Exemplo1: Pode-se criar na raiz o diretório site/home/views para armazenar e organizar os arquivos de visão nesse diretório e site/home/controles para os arquivos de controle.

Exemplo2: Pode-se criar um diretório docs para armazenamento de arquivos.

Exemplo3: Pode-se criar um diretório adm para implementar e organizar classes de construção de um gerenciador.

### 3. Classes MVC

#### 3.1. Classes Model (Modelo)

As classes de modelo são classes que representam a lógica de programação. Cada classe de modelo obrigatoriamente possui uma classe de repositório com a nomenclatura Repos[nome da classe] e uma tabela no banco com a nomenclatura tb\_[nome da classe]. Também devem herdar da superclasse AppModelo.

Importante: todos os atributos da tabela do BD de uma classe modelo devem ter os mesmos nomes dos atributos da classe.

Ex: Associado.class.php é o arquivo onde está implementada a classe modelo Associado que herda de AppModelo.

Supondo que a classe Associado tenha os atributos \$nome, \$fone e \$email sua tabela correspondente no banco de dados (tb\_associado) deve ter os atributos nome, fone e email (não necessariamente na mesma ordem).

O repositório da classe Associado deve ser a classe ReposAssociado que herda de Repos.class.php e devem ser armazenadas no diretório framework/global/app.modelo/repositórios.

Passa em seu construtor uma id (facultativa) caso queira instanciar um objeto já preenchido com os atributos do banco de dados.

[Ver descrição das funções em global/app.modelo/AppModelo.class.php].

#### 3.2. Classes Views (Visão)

As classes de visão são responsáveis pela manipulação, montagem e escrita na tela.

Essas classes iniciam com o prefixo Vs seguido do nome (Vs[nome da classe]). Todas as classes views herdam de global/app.visual/AppView.class.php e obrigatoriamente possuem um arquivo html correspondente com mesmo nome (a primeira letra deve ser minúscula).

Cada view também pode ter uma classe de controle, um arquivo css, um arquivo js, uma view de cabeçalho ou header (que será montada acima) e uma view de rodapé ou footer (que será montada abaixo). Todos esses parâmetros deverão ser indicados na instanciação da classe. No construct também deve-se informar o diretório da view.

[Ver descrição das funções em global/app.visual/AppView.class.php]



### 3.3. Classes Control (Controle)

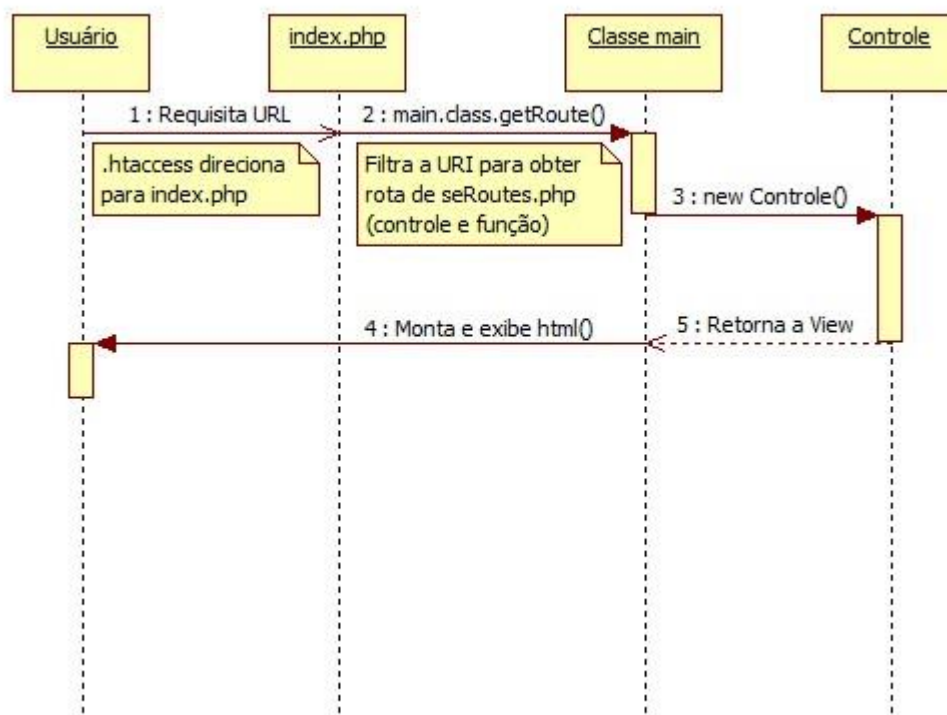
As classes de controle são responsáveis por fazer as requisições e intermediações entre o modelo e as views. Sempre herdam de `global/app.controle/AppCtrl.class.php`.

[Ver descrição das funções em `global/app.controle/AppCtrl.class.php`]

## 4. Sequência de Funcionamento - `index.php` e `.htaccess`

O arquivo `.htaccess` é definido para sempre chamar o `index.php` através do código html 404. O `index.php` é responsável por montar o corpo do html e inserir as views de acordo com a url chamada. Além disso define os scripts, instancia a classe `Global.class.php` e chama o controle de acordo com a url especificada (ver `framework/config/setRoutes.php`).

O diagrama de sequência abaixo representa o processamento geral do framework.



## 5. Exemplos

### 5.1. VsExemploHome.class.php

```

/**
 *
 * Classe de view
 * Herda de AppView
 * @param Array $params com os valores passados via post ou get por Global.js->showVs()
 *
 */
class VsExemploHome extends AppView {

    function __construct($params = false) {
        # O parent::__construct recebe um boolean para indicar se deve criar o controle,
        # Diretório do arquivo,
        # se possui css,
        # se possui js,
        # qual header deve ser carregada ou false,
        # qual footer deve ser carregado ou false

        # O controle, o html, o css e o js devem seguir as nomeclaturas CtrlExemploHome,
        exemploHome.html, exemploHome.css e exemploHome.js respectivamente

        # parent::__construct();
        parent::__construct (true, 'site/home', true, true, 'VsHeader', 'VsFooter');

        # Se é rastreado pelo google analytics (Global.js->trackGoogleAnalytics(url))
        # O script é criado na função AppView->show();
        $this->trackGoogle = true;

        # Obtendo um elemento div de exemploHome.html pela id
        $div1 = $this->getHtmlTag('div1');

        #Inserindo conteúdo em um elemento
        $div1->setIn('string');
        $div1->setIn('string', true); // True indica que todo o conteúdo já existente deve ser
        substituído

        #Criando um elemento html com a biblioteca DHtml (ver outras classes em
        framework/lib/dhtml)
        $img = new DImg('alt', 'src');

        #Inserindo o objeto DImg na div criada
        $div1->setIn($img);

        #Inserindo o objeto DImg na div criada em formato string
        $div1->setIn($img->toStr(). '<br />');

        # Uma elemento html também pode ser obtido pelo valor de um atributo
        // Retorna a primeira div cujo valor do atributo name é igual a 'div2'
        $div2 = $this->getHtmlTagByAtt('div', 'name', 'div2', 1);

        #Executa um comando ou função js
        //@param comando ou função $jsfunc
        //@param number $pos: indica se executa antes (0) ou depois (1) do load (default = 0)
        $this->callLoadJs('alert("msg")', 0);

        #Definindo atributos de um elemento
        $button = $this->getHtmlTag('button');
    }
}

```

```
$button->onclick = 'alert("click");'
```

```
$img->alt = 'logo';
```

```
$img->src = Glb::$CONFIG['URL'].'global/app.visual/imagens/logo.png';
```

```
// Glb::$CONFIG['URL']: url global definida em framework/config/setGlobalVars.php
```

```
#Chamando funções de controles
```

```
$button->onclick = AppCtrl::callCtrlPost(
    '[id do container]',
    '[nome da classe controle]',
    '[nome da função do controle]',
    $array_com_os_parametros_da_funcao,
    'mensagem do loading'
);
```

```
#Submetendo um form para um container
```

```
// A função deve receber um parâmetro $post
```

```
$button->onclick = AppCtrl::submitForm(
    '[id do container]',
    '[nome da classe controle]',
    '[nome da função do controle]',
    '[nome do form]',
    'mensagem do loading'
);
```

```
#Submetendo um form com input file para um container
```

```
// A função deve receber um parâmetro $post
```

```
$button->onclick = AppCtrl::formUpload(
    '[id do container]',
    '[nome da classe controle]',
    '[nome da função do controle]',
    '[nome do form]',
    'mensagem do loading'
);
}
}
```

## 5.2. ExemploCtrlHome.class.php

```

/**
 * Classe de controle
 * Herda de AppCtrl
 */
class CtrlExemploHome extends AppCtrl {
    function __construct() {
        parent::__construct();
    }

    # Supondo acesso à www.site.com.br/exemplo-home
    # o fluxo do framework acessará setRoutes.php e
    # verificará se exemplo-home está definido como rota
    (exemplo-home => 'CtrlExemploHome[showView()]')
    # showView() retorna VsExemploHome().

    # Qualquer view pode ser chamada de um controle.
    # Uma função de controle pode também receber parâmetros pela url informada.
    # Supondo acesso à www.site.com.br/home/10/x
    # showView() instancia e retorna VsExemploHome($param1, $param2)
    # onde $param1 = 10 e $param2 = 'x'

    # Uma função de controle pode ser requisitada através de AppCtrl::callCtrlPost()
    # Os parâmetros devem seguir o array passado por callCtrlPost
    # (ver global/app.controle/AppCtrl.class.php)

    # Ex: Imprimindo uma view em um container chamado por
    # AppCtrl::callCtrlPost(
        'div',
        'CtrlExemploHome',
        'showVsExemplo2',
        array('value' => 'xx'),
        'loading'
    );
    function showVsExemplo2($value) {
        // $value recebe 'xx'
        $view = new VsExemplo2($value);
        $view->show();
    }
}

```

### 5.3. ExemploModelo.class.php

```

/**
 * Classe herdada de AppModelo.
 * Os atributos devem ser iguais ao da tabela no BD e sempre deve ter o atributo id.
 * Para chamar o repositório dessa classe: NomeClasse::BD(tipo,chave,aux). Ver em
 AppModelo.class.
 * Toda classe modelo possui um repositório Repos[Nome da classe] e uma tabela no banco
 tb_[nome da classe] (tudo lowercase)
 * Tipo: string select, update, insert ou delete
 * Chave: int número da query a ser usada no repositório
 * Aux: Variáveis passadas para utilização na query (No caso de insert = obj)
 * @param int id do obj no BD para carregar o objeto na instância (facultativo)
 * @method get: Herdado de AppModelo - Retorna um atributo
 * @method set: Herdado de AppModelo - Define um atributo
 * @method getObj: Herdado de AppModelo - Retorna um obj do BD de acordo com o
 parâmetro (array).
 * @method setObj: Herdado de AppModelo - Define todos os atributos de um objeto com os
 valores do parâmetro (array).
 * @method insert: Herdado de AppModelo - Insere um obj no BD
 * @method update: Herdado de AppModelo - Atualiza um obj no BD
 * @method delete: Herdado de AppModelo - Deleta um obj do BD
 * @method deleteAll: Herdado de AppModelo (static) - Deleta todos os objs do BD
 * @method BD (tipo, chave, aux): Herdado de AppModelo (static) - Acessa o repositório do
 Obj (app.modelo->Repos[nome da classe].class.php)
 * @author Diogo (d.bezerra@yahoo.com.br)
 */
class ExemploModelo extends AppModelo {
    public $id;
    public $nome;
    public $fone;
    public $email;

    function __construct($id = NULL) {
        parent::__construct ( $id );
    }
}

# Exemplo
function enviaEmail($id, $email) {
    # Cria um objeto com os atributos obtidos do banco de dados
    $exMod = new ExemploModelo($id);

    #Cria um objeto "vazio"
    $exMod = new ExemploModelo();

    # Definindo atributo
    $exMod->set('nome', '[string nome]');
    $exMod->set('email', $email);

    # Obtendo valores dos atributos
    $destino = $exMod->get('email');

    # Instanciando o plugin Email (framework/plugins/email)
    $e = new Email($remetente, $destino, $assunto, $texto);
    $e->enviar(); // envia o email

    # Atualizando objeto no banco
    $exMod->update(); // Somente os atributos definidos serão atualizados

```

```
# Inserindo um objeto no banco
$exMod->set('nome', $novo_valor);
$exMod->set('fone', $novo_valor);
$exMod->set('email', $novo_valor);
$exMod->insert();

# O repositório pode ser acessado diretamente para executar qualquer query
$rs = ExemploModelo::BD('select', 0, array('nome'=>$nome)); // Retorna objetos
cujo nome = $nome
// Retorna falso se não houver resultados ou o fetch da consulta
if($rs) {
    foreach ($rs as $obj) {
        $nome = $obj['nome'];
        $fone = $obj['fone'];
        $email = $obj['email'];

        # Os atributos do objeto modelo podem ser definidos
        # diretamente através do array retornado
        $exMod->setObj($obj);
    }
}
}
```

#### 5.4. ReposExemploModelo.class.php

[Ver global/app.modelo/repositórios/ReposExemploModelo.class.php]

#### 6. Plugins

...